Weighted Model Counting Without Parameter Variables

No Author Given

No Institute Given

Abstract. The abstract should briefly summarize the contents of the paper in 150–250 words.

Keywords: First keyword · Second keyword · Another keyword.

1 Introduction

Related Work. Many WMC inference algorithms such as Ace[1], c2d [10], and miniC2D [14] work by compilation to tractable representations such as arithmetic circuits, deterministic, decomposable negation normal form [8], and sentential decision diagrams (SDDs) [11]. Some attempts have previously been made to skip this intermediate stage of representing a Bayesian network as a WMC instance and compile directly to a more convenient representation. Specifically, direct compilation from Bayesian networks to SDDs [7] and from structured Bayesian networks (i.e., a generalisation of Bayesian networks) to probabilistic SDDs [17] have been considered. To the best of the authors' knowledge, neither compilation approach has a publicly available implementation.

Our work is similar in that it introduces a new representation for computational problems that is based on pseudo-Boolean functions—pseudo-Boolean projection (PBP). We formally show that every WMC problem instance has a corresponding PBP instance. We argue that this new format is more appropriate for WMC solvers whose execution is build on pseudo-Boolean function manipulation, i.e., ADDMC [12] and DPMC [13]. Finally, we show how most methods of encoding probabilistic inference on Bayesian networks to WMC can be adapted to the PBP format. This transformation significantly improves inference speed, outperforming all other considered algorithm-encoding combinations.

Notes.

- we also show correctness
- and introduce a new notation

---

[1] http://reasoning.cs.ucla.edu/ace/

2   Weighted Model Counting

TODO: Mention that formulas, clauses, and models are all treated as sets.

Definition 1 (WMC). *A* WMC instance *is a tuple* (ϕ, X_I, X_P, w)*, where* X_I *is the set of indicator variables,* X_P *is the set of parameter variables (with* $X_I \cap X_P = \emptyset$*),* ϕ *is a propositional formula in CNF over* $X_I \cup X_P$*, and* $w \colon X_I \cup X_P \cup \{\neg x \mid x \in X_I \cup X_P\} \to \mathbb{R}$ *is the weight function. The* answer *of the instance is* $\sum_{Y \models \phi} \prod_{Y \models l} w(l)$.

Definition 2 (Minimum-Cardinality WMC). *A* minimum-cardinality WMC *instance has same tuple as a WMC instance, but its answer is defined to be* $\sum_{Y \models \phi, \ |Y|=k} \prod_{Y \models l} w(l)$*, where* $k = \min_{Y \models \phi, \ Y \neq \emptyset} |Y|$*, if* k *exists, otherwise the answer is zero.*

Our definition of WMC is largely based on the standard definition [5], but explicitly partitions variables into indicator and parameter variables. In practice, we identify this partition in one of two ways. If an encoding is generated by Ace, then variable types are explicitly identified in the LMAP file generated alongside the encoding. Otherwise, we define a variable x to be a parameter variable if $w(x) = w(\neg x) = 1$.

2.1   Bayesian Network Encodings (mostly done)

A *Bayesian network* is a directed acyclic graph with random variables as vertices and edges as conditional dependencies. As is common in related literature [9, 16], we assume that each variable has a finite number of values. We call a Bayesian network *binary* if every variable has two values. If all variables have finite numbers of values, the probability function associated with each variable v can be represented as a *conditional probability table* (CPT), i.e., a table with a row for each combination of values that v and its parent vertices can take. Each row then also has a *probability*, i.e., a number in $[0, 1]$.

WMC is a well-established technique for Bayesian network inference, particularly effective on networks where most variables have only a few possible values [9]. Many ways of encoding a Bayesian network into a WMC instance have been proposed. We will refer to them based on initials of the authors and the year of publication. Darwiche was the first to suggest the d02 [9] encoding that, in many ways, remains the foundation behind most other encodings. He also introduced the distinction between *indicator* and *parameter variables*; the former represent variable-value pairs in the Bayesian network, while the latter are associated with probabilities in the CPTs. The encoding sbk05 [16] is the only encoding that deviates from this arrangement: for each variable in the Bayesian network, one indicator variable acts simultaneously as a parameter variable. Chavira and Darwiche propose cd05 [3] where they shift from WMC to minimum-cardinality WMC because that allows the encoding to have fewer variables and clauses. In particular, they propose a way to use the same parameter variable to represent all probabilities in a CPT that are equal and keep only

clauses that 'imply' parameter variables (i.e., omit clauses where a parameter variable implies indicator variables). In their next encoding, `cd06` [4], the same authors optimise the aforementioned implication clauses, choosing the smallest sufficient selection of indicator variables. A decade later, Bart et al. present `bklm16` [2] that improves upon `cd06` in two ways. First, they optimise the number of indicator variables used per Bayesian network variable from a linear to a logarithmic amount. Second, they introduce a scaling factor that can 'absorb' one probability per Bayesian network variable. However, for this work, we choose to disable the latter improvement since this scaling factor is often small enough to be indistinguishable from zero without the use of arbitrary precision arithmetic. Indeed, even a small Bayesian network with seven mutually independent binary variables, 0.1 and 0.9 probabilities each, is already big enough for the scaling factor to be exactly equal to zero (as produced by the `bklm16` encoder[2]).

3   Pseudo-Boolean Functions

Notation. For any propositional formula ϕ over a set of variables X and $p, q \in \mathbb{R}$, let $[\phi]_q^p \colon 2^X \to \mathbb{R}$ be a pseudo-Boolean function defined as

$$[\phi]_q^p(Y) := \begin{cases} p & \text{if } Y \models \phi \\ q & \text{otherwise} \end{cases}$$

for any $Y \subseteq X$.

Definition 3 (Operations). *Let $f, g \colon 2^X \to \mathbb{R}$ be pseudo-Boolean functions, $x, y \in X$, $Y = \{y_i\}_{i=1}^n \subseteq X$, and $r \in \mathbb{R}$. Operations such as addition and multiplication are defined pointwise as*

$$(f + g)(Y) := f(Y) + g(Y), \quad and \quad (f \cdot g)(Y) := f(Y) \cdot g(Y).$$

Note that this means that binary operations on pseudo-Boolean functions inherit properties such as associativity and commutativity. By not distinguishing between a real number and a pseudo-Boolean function that always returns that number, we can use the same definitions to define scalar *operations as*

$$(r + f)(Y) = r + f(Y), \quad and \quad (r \cdot f)(Y) = r \cdot f(Y).$$

Restrictions $f|_{x=0}, f|_{x=1} \colon 2^X \to \mathbb{R}$ of f are defined as

$$f|_{x=0}(Y) := f(Y \setminus \{x\}), \quad and \quad f|_{x=1}(Y) := f(Y \cup \{x\})$$

for all $Y \subseteq X$.

Projection $\exists_x$ is an endomorphism $\exists_x \colon \mathbb{R}^{2^X} \to \mathbb{R}^{2^X}$ defined as

$$\exists_x f := f|_{x=1} + f|_{x=0}.$$

Since projection is commutative (i.e., $\exists_x \exists_y f = \exists_y \exists_x f$) [12, 13], we can define $\exists_Y \colon \mathbb{R}^{2^X} \to \mathbb{R}^{2^X}$ as $\exists_Y := \exists_{y_1} \exists_{y_2} \ldots \exists_{y_n}$. Throughout the paper, projection is assumed to have the lowest precedence (e.g., $\exists_x fg = \exists_x(fg)$).

---

[2] http://www.cril.univ-artois.fr/kc/bn2cnf.html

Below we list some properties of the operations on pseudo-Boolean functions discussed in this section that can be conveniently represented using our syntax. The proofs of all these properties follow directly from the definitions.

Proposition 1 (Basic Properties). *For any propositional formulas ϕ and ψ, and $a, b, c, d \in \mathbb{R}$,*

- $[\phi]_b^a = [\neg\phi]_a^b$;
- $c + [\phi]_b^a = [\phi]_{b+c}^{a+c}$;
- $c \cdot [\phi]_b^a = [\phi]_{bc}^{ac}$;
- $[\phi]_b^a \cdot [\phi]_d^c = [\phi]_{bd}^{ac}$;
- $[\phi]_0^1 \cdot [\psi]_0^1 = [\phi \wedge \psi]_0^1$.

And for any pair of pseudo-Boolean functions $f, g\colon 2^X \to \mathbb{R}$ and $x \in X$, $(fg)|_{x=i} = f|_{x=i} \cdot g|_{x=i}$ for $i = 0, 1$.

4   PBP as a Computational Problem

Definition 4 (PBP Instance). *A PBP instance is a tuple (F, X, ω), where X is the set of variables, F is a set of pseudo-Boolean functions $2^X \to \mathbb{R}$, and $\omega \in \mathbb{R}$ is the scaling factor.*[3] *The* answer *of the instance is $\omega \cdot \left(\exists_X \prod_{f \in F} f \right) (\emptyset)$.*

Remark 1. Adding scaling factor ω to the definition allows us to remove clauses that consist entirely of a single parameter variable. The idea of extracting some of the structure of the WMC instance into an external multiplicative factor originates in one of the Bayesian network encodings [2].

4.1   From WMC to PBP

Notes.

- The algorithm is based on several observations that will be made precise in Theorem 2. First, all weights except for $\{w(p) \mid p \in X_P\}$ are redundant as they either duplicate an already-defined weight or are equal to one. Second, each clause has at most one parameter variable. Third, if the parameter variable is negated, we can ignore the clause (this idea first appears in the cd05 paper [3]).
- We formulate our algorithm as a sequel to the WMC encoding procedure because the implementations of Bayesian network WMC encodings are all closed-source. However, as all transformations in Algorithm 1 are local, the algorithm can be efficiently incorporated into a WMC encoding algorithm with no noticeable slowdown.

---

[3] We will further restrict F to only contain functions that have exactly two values. While this restriction is not necessary from a computational point of view, such functions can be conveniently represented both in text files and using the syntax proposed in this paper.

Algorithm 1: WMC to PBP transformation

Data: WMC instance (ϕ, X_I, X_P, w)
Result: PBP instance (F, X, ω)

1 $F \leftarrow \emptyset$;
2 $\omega \leftarrow 1$;
3 let $d\colon X_P \to \mathbb{N}$ be defined as $p \mapsto |\{o \in X_P \mid o \le p\}|$;
4 **foreach** *clause* $c \in \phi$ **do**
5   **if** $c \cap X_P = \{p\}$ *for some* p **and** $w(p) \ne 1$ **then**
6   **if** $|c| = 1$ **then**
7 $\omega \leftarrow \omega \times w(p)$;
8   **else**
9 $F \leftarrow F \cup \left\{ \left[\bigwedge_{l \in c \setminus \{p\}} \neg l \right]_1^{w(p)} \right\}$;
10   **else if** $\{p \mid \neg p \in c\} \cap X_P = \emptyset$ **then**
11 $F \leftarrow F \cup \{[c]_0^1\}$;
12 **foreach** *indicator variable* $v \in X_I$ **do**
13   **if** $\{[v]_1^p, [\neg v]_1^q\} \subseteq F$ *for some* p *and* q **then**
14 $F \leftarrow F \setminus \{[v]_1^p, [\neg v]_1^q\} \cup \{[v]_q^p\}$;
15 replace every variable v in F with `rename(v)`;
16 **return** (F, X_I, ω);
17 **Function** `rename(v)`:
18 $S \leftarrow \{u \in X_P \mid u \le v\}$;
19   **if** $S = \emptyset$ **then return** v;
20   **return** $v - d(\max S)$;

- No obvious way to do this for `sbk05` because the roles of indicator and parameter (i.e., 'chance') variables overlap [16].
- Benefits of having this proof in the paper:
  - It puts all encodings on a common ground.
  - It illustrates the convenience of our notation for reasoning about (certain types of) pseudo-Boolean functions.
  - It's too big and too important to be left for the appendix.

4.2 Proof of Correctness

Notes.

- This just says that p is equivalent to a conjunction.
- The first group of conditions applies to `d02`, while the second group applies to `bklm16`.
- For `cd05` and `cd06`, condition *2bi* should be replaced with $w(\neg p) = 1$.

TODO

- check if each condition is actually used. Maybe turn this into a paragraph that gives an overview of the proof.
- condition 1 is necessary in both cases because we're ignoring the weights of indicator variables (explicitly acknowledge this)
- add the `cd05`/`cd06` correctness theorem

Theorem 1 (Early Projection [12, 13], verbatim). *Let X and Y be sets of variables. For all functions $f\colon 2^X \to \mathbb{R}$ and $g\colon 2^Y \to \mathbb{R}$, if $x \in X \setminus Y$, then $\exists_x(f \cdot g) = (\exists_x f) \cdot g$.*

Lemma 1. *For any pseudo-Boolean function $f\colon 2^X \to \mathbb{R}$,*

$$(\exists_X f)(\emptyset) = \sum_{Y \subseteq X} f(Y).$$

Proof. If $X = \{x\}$, then

$$(\exists_x f)(\emptyset) = (f|_{x=1} + f|_{x=0})(\emptyset) = f|_{x=1}(\emptyset) + f|_{x=0}(\emptyset) = \sum_{Y \subseteq \{x\}} f(Y).$$

This easily extends to $|X| > 1$ by the definition of projection on sets of variables.

Proposition 2. *Let (ϕ, X_I, X_P, w) be a WMC instance. Then*

$$\left(\left\{ [c]_0^1 \mid c \in \phi \right\} \cup \left\{ [x]_{w(\neg x)}^{w(x)} \;\middle|\; x \in X_I \cup X_P \right\}, X_I \cup X_P, 1 \right) \tag{1}$$

is a PBP instance with the same answer.

Proof. Let $f = \prod_{c \in \phi} [c]_0^1$, and $g = \prod_{x \in X_I \cup X_P} [x]_{w(\neg x)}^{w(x)}$. Then the answer to the WMC instance (1) is

$$(\exists_{X_I \cup X_P} fg)(\emptyset) = \sum_{Y \subseteq X_I \cup X_P} (fg)(Y) = \sum_{Y \subseteq X_I \cup X_P} f(Y)g(Y)$$

by Lemma 1. Note that

$$f(Y) = \begin{cases} 1 & \text{if } Y \models \phi, \\ 0 & \text{otherwise,} \end{cases} \quad \text{and} \quad g(Y) = \prod_{Y \models l} w(l),$$

which means that

$$\sum_{Y \subseteq X_I \cup X_P} f(Y)g(Y) = \sum_{Y \models \phi} \prod_{Y \models l} w(l)$$

as required.

Theorem 2 (Correctness). *Algorithm 1, when given a WMC instance (ϕ, X_I, X_P, w), returns PBP instance with the same answer, provided the following conditions are satisfied:*

1. *for all indicator variables $i \in X_I$, $w(i) = w(\neg i) = 1$,*
2. *and either*
   (a) *for all parameter variables $p \in X_P$, there is a non-empty family of literals $(l_i)_{i=1}^n$ such that*
      i. *$w(\neg p) = 1$,*
      ii. *$l_i \in X_I$ or $\neg l_i \in X_I$ for all $i = 1, \ldots, n$,*
      iii. *and $\{c \in \phi \mid p \in c \text{ or } \neg p \in c\} = \{p \vee \bigvee_{i=1}^n \neg l_i\} \cup \{l_i \vee \neg p \mid i = 1, \ldots, n\}$;*
   (b) *or for all parameter variables $p \in X_P$,*
      i. *$w(p) + w(\neg p) = 1$,*
      ii. *for any clause $c \in \phi$, $|c \cap X_P| \leq 1$,*
      iii. *there is no clause $c \in \phi$ such that $\neg p \in c$,*
      iv. *if $\{p\} \in \phi$, then there is no clause $c \in \phi$ such that $c \neq \{p\}$ and $p \in c$,*

      v. *and for any $c, d \in \phi$ such that $c \neq d$, $p \in c$ and $p \in d$, $\bigwedge_{l \in c \setminus \{p\}} \neg l \wedge \bigwedge_{l \in d \setminus \{p\}} \neg l$ is false.*

Proof. By Proposition 2,

$$\left(\{[c]_0^1 \mid c \in \phi\} \cup \left\{ [x]_{w(\neg x)}^{w(x)} \;\middle|\; x \in X_I \cup X_P \right\}, X_I \cup X_P, 1 \right) \tag{2}$$

is a PBP instance with the same answer as the given WMC instance. By Definition 4, its answer is

$$\left(\exists_{X_I \cup X_P} \left(\prod_{c \in \phi} [c]_0^1 \right) \prod_{x \in X_I \cup X_P} [x]_{w(\neg x)}^{w(x)} \right) (\emptyset) \tag{3}$$

Since both Conditions 2a and 2b ensure that each clause in ϕ has at most one parameter variable, we can partition ϕ into $\phi_* := \{c \in \phi \mid \mathtt{Vars}(c) \cap X_P = \emptyset\}$ and $\phi_p := \{c \in \phi \mid \mathtt{Vars}(c) \cap X_P = \{p\}\}$ for all $p \in X_P$. We can then use Theorem 1 to reorder (3) into

$$\left(\exists_{X_I} \left(\prod_{x \in X_I} [x]_{w(\neg x)}^{w(x)} \right) \left(\prod_{c \in \phi_*} [c]_0^1 \right) \prod_{p \in X_P} \exists_p [p]_{w(\neg p)}^{w(p)} \prod_{c \in \phi_p} [c]_0^1 \right) (\emptyset).$$

Let us first consider how the unfinished WMC instance (F, X_I, ω) after the loop on Lines 4 to 11 differs from (2). Note that Algorithm 1 leaves each $c \in \phi_*$ unchanged, i.e., adds $[c]_0^1$ to F. We can then fix an arbitrary $p \in X_P$ and let F_p be the set of functions added to F as a replacement of ϕ_p. It is sufficient to show that

$$\omega \prod_{f \in F_p} f = \exists_p [p]_{w(\neg p)}^{w(p)} \prod_{c \in \phi_p} [c]_0^1. \tag{4}$$

Note that under Condition 2a,

$$\bigwedge_{c \in \phi_p} c \equiv p \Leftrightarrow \bigwedge_{i=1}^{n} l_i$$

for some family of indicator variable literals $(l_i)_{i=1}^{n}$. Thus,

$$\exists_p [p]_{w(\neg p)}^{w(p)} \prod_{c \in \phi_p} [c]_0^1 = \exists_p [p]_1^{w(p)} \left[p \Leftrightarrow \bigwedge_{i=1}^{n} l_i \right]_0^1.$$

If $w(p) = 1$, then

$$\exists_p [p]_1^{w(p)} \left[p \Leftrightarrow \bigwedge_{i=1}^{n} l_i \right]_0^1 = \left[p \Leftrightarrow \bigwedge_{i=1}^{n} l_i \right]_0^1 \Bigg|_{p=1} + \left[p \Leftrightarrow \bigwedge_{i=1}^{n} l_i \right]_0^1 \Bigg|_{p=0}. \quad (5)$$

Since for any input, $\bigwedge_{i=1}^{n} l_i$ is either true or false, exactly one of the two summands in Eq. (5) will be equal to one, and the other will be equal to zero, and so

$$\left[p \Leftrightarrow \bigwedge_{i=1}^{n} l_i \right]_0^1 \Bigg|_{p=1} + \left[p \Leftrightarrow \bigwedge_{i=1}^{n} l_i \right]_0^1 \Bigg|_{p=0} = 1,$$

where 1 is a pseudo-Boolean function that always returns one. On the other side of Eq. (4), since $F_p = \emptyset$, and ω is unchanged, we get $\omega \prod_{f \in F_p} f = 1$, and so Eq. (4) is satisfied under Condition 2a when $w(p) = 1$.

If $w(p) \neq 1$, then

$$F_p = \left\{ \left[\bigwedge_{i=1}^{n} l_i \right]_1^{w(p)} \right\},$$

and $\omega = 1$, and so we want to show that

$$\left[\bigwedge_{i=1}^{n} l_i \right]_1^{w(p)} = \exists_p [p]_1^{w(p)} \left[p \Leftrightarrow \bigwedge_{i=1}^{n} l_i \right]_0^1,$$

and indeed

$$\exists_p [p]_1^{w(p)} \left[p \Leftrightarrow \bigwedge_{i=1}^{n} l_i \right]_0^1 = w(p) \cdot \left[\bigwedge_{i=1}^{n} l_i \right]_0^1 + \left[\bigwedge_{i=1}^{n} l_i \right]_1^0 = \left[\bigwedge_{i=1}^{n} l_i \right]_1^{w(p)}.$$

This finishes the proof of the correctness of the first 'foreach' loop under Condition 2a.

Now let us assume Condition 2b. We still want to prove Eq. (4). If $w(p) = 1$, then $F_p = \emptyset$, and $\omega = 1$, and so the left-hand side of Eq. (4) is equal to one.

Then the right-hand side is

$$\exists_p[p]_0^1 \prod_{c\in\phi_p} [c]_0^1 = \exists_p \left[p \wedge \bigwedge_{c\in\phi_p} c\right]_0^1 = \exists_p[p]_0^1 = 0 + 1 = 1$$

since $p \in c$ for every clause $c \in \phi_p$.

If $w(p) \neq 1$, and $\{p\} \in \phi_p$, then, by Condition 2(b)iv, $\phi_p = \{\{p\}\}$, and Algorithm 1 produces $F_p = \emptyset$ and $\omega = w(p)$, and so

$$\exists_p[p]_{w(\neg p)}^{w(p)}[p]_0^1 = \exists_p[p]_0^{w(p)} = w(p) = \omega \prod_{f\in F_p} f.$$

The only remaining case is when $w(p) \neq 1$ and $\{p\} \notin \phi_p$. Then $\omega = 1$, and

$$F_p = \left\{ \left[\bigwedge_{l\in c\setminus\{p\}} \neg l\right]_1^{w(p)} \;\middle|\; c \in \phi_p\right\},$$

so need to show that

$$\prod_{c\in\phi_p} \left[\bigwedge_{l\in c\setminus\{p\}} \neg l\right]_1^{w(p)} = \exists_p[p]_{1-w(p)}^{w(p)} \prod_{c\in\phi_p} [c]_0^1.$$

We can rearrange the right-hand side as

$$\exists_p[p]_{1-w(p)}^{w(p)} \prod_{c\in\phi_p} [c]_0^1 = \exists_p[p]_{1-w(p)}^{w(p)} \left[\bigwedge_{c\in\phi_p} c\right]_0^1 = \exists_p[p]_{1-w(p)}^{w(p)} \left[p \vee \bigwedge_{c\in\phi_p} c\setminus\{p\}\right]_0^1$$

$$= w(p) + (1-w(p)) \left[\bigwedge_{c\in\phi_p} c\setminus\{p\}\right]_0^1 = w(p) + \left[\bigwedge_{c\in\phi_p} c\setminus\{p\}\right]_0^{1-w(p)}$$

$$= \left[\bigwedge_{c\in\phi_p} c\setminus\{p\}\right]_{w(p)}^1 = \left[\neg \bigwedge_{c\in\phi_p} c\setminus\{p\}\right]_1^{w(p)} = \left[\bigvee_{c\in\phi_p} \neg(c\setminus\{p\})\right]_1^{w(p)}$$

$$= \left[\bigvee_{c\in\phi_p} \neg \bigvee_{l\in c\setminus\{p\}} l\right]_1^{w(p)} = \left[\bigvee_{c\in\phi_p}\bigwedge_{l\in c\setminus\{p\}} \neg l\right]_1^{w(p)}.$$

By Condition 2(b)v, $\bigwedge_{l\in c\setminus\{p\}} \neg l$ can be true for at most one $c \in \phi_p$, and so

$$\left[\bigvee_{c\in\phi_p}\bigwedge_{l\in c\setminus\{p\}} \neg l\right]_1^{w(p)} = \prod_{c\in\phi_p} \left[\bigwedge_{l\in c\setminus\{p\}} \neg l\right]_1^{w(p)}$$

which is exactly what we needed to show. This ends the proof that the first loop of Algorithm 1 preserves the answer under both Condition 2a and Condition 2b. Finally, the loop on Lines 12 to 14 of Algorithm 1 replaces $[v]_1^p[\neg v]_1^q$ with $[v]_q^p$ (for some $v \in X_I$ and $p, q \in \mathbb{R}$), but, of course,

$$[v]_1^p[\neg v]_1^q = [v]_1^p[v]_q^1 = [v]_q^p,$$

i.e., the answer is unchanged.

5 Experimental Evaluation

We run a set of experiments, comparing all five original Bayesian network encodings (`bklm16`, `cd05`, `cd06`, `d02 sbk05`) as well as the first four with Algorithm 1 applied afterwards.[4] For each encoding `e`, we will write `e++` to denote the combination of encoding a Bayesian network as a WMC instance using `e` and transforming it into a PBP instance using Algorithm 1. Along with DPMC[5], we also include WMC algorithms used in the papers that introduce each encoding: Ace for `cd05`, `cd06`, and `d02`; Cachet[6] [15] for `sbk05`; and c2d[7] [10] with query-dnnf[8] for `bklm16`. We focus on the following questions:

- Can parameter variable elimination improve inference speed?
- How does DPMC combined with encodings without (and with) parameter variables compare with other WMC algorithms and other encodings?
- Which instances is our approach particularly successful on (compared to other algorithms and encodings and to the same encoding before our transformation)?
- What proportion of variables is typically removed?
- Do some encodings benefit from this transformation more than others?

5.1 Setup (mostly done)

DPMC is run with tree decomposition based planning and algebraic decision diagram based execution—the best-performing combination in the original set of experiments [13]. We use a single iteration of htd [1] to generate approximately optimal tree decompositions—we found that this configuration is efficient enough to handle huge instances, and yet the width of the returned decomposition is unlikely to differ from optimal by more than one or two. We also enabled DPMC's greedy mode. This mode (which was not part of the original paper [13]) optimises the order in which pseudo-Boolean functions are multiplied by prioritising functions with small representations.

For experimental data, we use Bayesian networks available with Ace and Cachet. We split them into the following groups:

---

[4] Recall that `cd05` and `cd06` are incompatible with DPMC.
[5] https://github.com/vardigroup/DPMC
[6] https://cs.rochester.edu/u/kautz/Cachet/
[7] http://reasoning.cs.ucla.edu/c2d/
[8] http://www.cril.univ-artois.fr/kc/d-DNNF-reasoner.html

- DQMR (390 instances) and
- Grid networks (450 instances) as described by Sang et al. [16];
- Mastermind (144 instances) and
- Random Blocks (256 instances) by Chavira et al. [6];
- other binary Bayesian networks (50 instances) including Plan Recognition [16], Friends and Smokers, Students and Professors [6], and `tcc4f`;
- non-binary classic networks (176 instances): `alarm`, `diabetes`, `hailfinder`, `mildew`, `munin1`–`4`, `pathfinder`, `pigs`, and `water`.

To perform Bayesian network inference with DPMC (or with any other WMC algorithm not based on compilation such as Cachet), one needs to select a probability to compute [13, 15]. If a network comes with an evidence file, we compute the probability of this evidence. Otherwise, let X be the variable last mentioned in the Bayesian network file. If `true` is one of the values of X, then we compute $\Pr(X = \text{true})$, otherwise we choose the first-mentioned value of X.

The experiments were run on a computing cluster with Intel Xeon E5-2630, Intel Xeon E7-4820, and Intel Xeon Gold 6138 processors with a 1000 s timeout separately on both encoding and inference, and a 32 GiB memory limit.[9]

5.2 Results

Notes.

- Figure 1 shows:
  - DPMC + `bklm16++` remains the best-performing combination across all time limits up to 1000 s with Ace + `cd06` and DPMC + `bklm16` not far behind;
  - DPMC + `bklm16++` to be 3.35 times faster than DPMC + `bklm16` and 2.96 times faster than Ace + `cd06`.
- Table 1 further shows that DPMC + `bklm16++` solves almost a hundred more instances than any other combination, and is the fastest in 69.1 % of them.
- TODO: now talk about Fig. 2.
- We also record numbers of variables in each encoding before and after our additional transformation.
- TODO: describe Fig. 3.
- Indeed, the numbers of variables in `bklm16++` and `d02++` encodings are equal for binary Bayesian networks (i.e., most of our data). Nonetheless, `bklm16++` performs much faster than `d02++`.

6 Conclusions

Notes.

- Benefits of my approach:

---

[9] Each instance was run on the same processor across all algorithms and encodings.
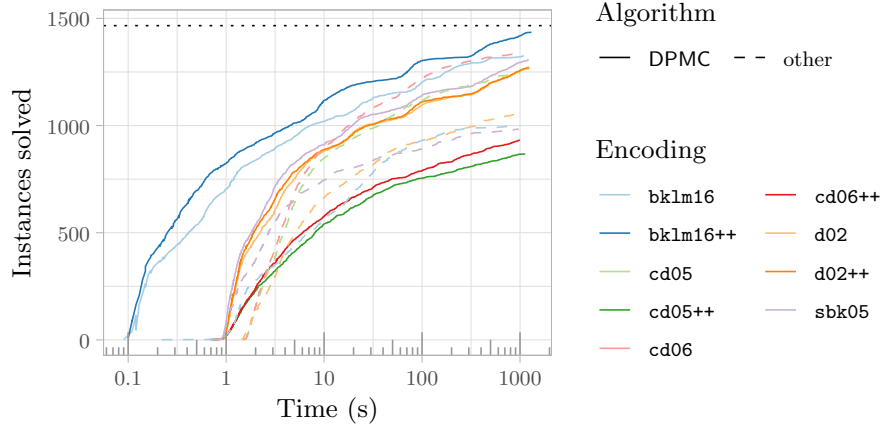
Fig. 1. Cumulative numbers of instances solved by each algorithm-encoding pair over time. The dotted line denotes the total number of instances used.
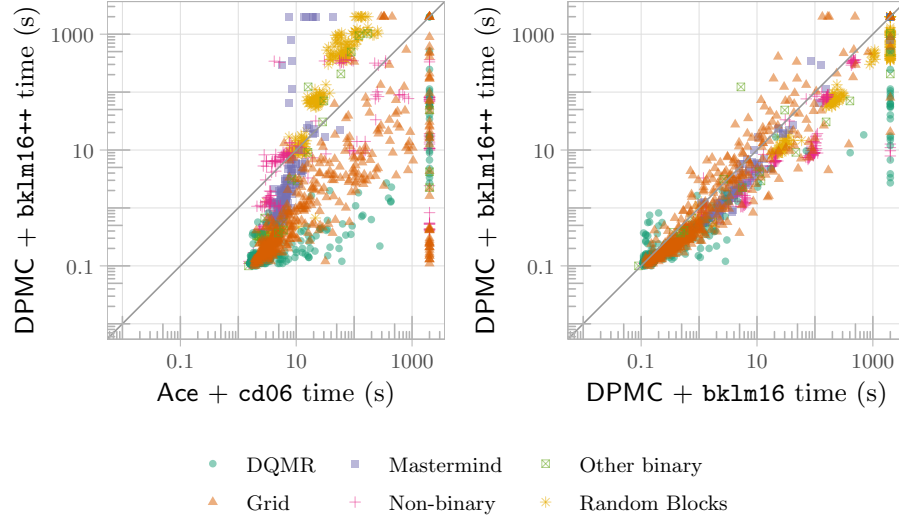

Fig. 2. An instance-by-instance comparison between DPMC+`bklm16++` (the best combination according to Fig. 1) and the second and third best performing combinations: Ace + `cd06` and DPMC + `bklm16`.
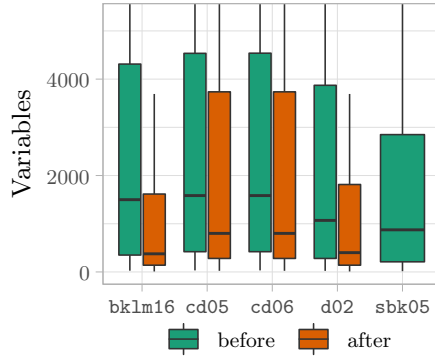
Fig. 3. Box plots of the numbers of variables in each encoding across all benchmark instances before and after applying Algorithm 1. Outliers and the top parts of some whiskers are omitted.

Table 1. The numbers of instances (out of 1466) that each algorithm and encoding combination solved faster than any other combination and in total.

Combination	Fastest	Solved
Ace + `cd05`	27	1247
Ace + `cd06`	135	1340
Ace + `d02`	56	1060
DPMC + `bklm16`	241	1327
DPMC + `bklm16++`	**992**	**1435**
DPMC + `cd05++`	0	867
DPMC + `cd06++`	0	932
DPMC + `d02`	1	1267
DPMC + `d02++`	7	1272
DPMC + `sbk05`	31	1308
c2d + `bklm16`	0	997
Cachet + `sbk05`	49	983

- smaller primal graph, so easier to perform tree decomposition
- Variable order is less likely to be obstructed by all the unnecessary 'parameter' variables.
- There are others, but they're not that important.
- Future work: do other WMC encodings also have this variable partition property?
- Do I need to formally consider extending a pseudo-Boolean function to a bigger domain?

References

1. Abseher, M., Musliu, N., Woltran, S.: htd - A free, open-source framework for (customized) tree decompositions and beyond. In: Salvagnin, D., Lombardi, M. (eds.) Integration of AI and OR Techniques in Constraint Programming - 14th International Conference, CPAIOR 2017, Padua, Italy, June 5-8, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10335, pp. 376–386. Springer (2017). https://doi.org/10.1007/978-3-319-59776-8_30
2. Bart, A., Koriche, F., Lagniez, J., Marquis, P.: An improved CNF encoding scheme for probabilistic inference. In: Kaminka, G.A., Fox, M., Bouquet, P., Hüllermeier, E., Dignum, V., Dignum, F., van Harmelen, F. (eds.) ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016). Frontiers in Artificial Intelligence and Applications, vol. 285, pp. 613–621. IOS Press (2016). https://doi.org/10.3233/978-1-61499-672-9-613
3. Chavira, M., Darwiche, A.: Compiling Bayesian networks with local structure. In: Kaelbling, L.P., Saffiotti, A. (eds.) IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland,

UK, July 30 - August 5, 2005. pp. 1306–1312. Professional Book Center (2005), http://ijcai.org/Proceedings/05/Papers/0931.pdf

4. Chavira, M., Darwiche, A.: Encoding CNFs to empower component analysis. In: Biere, A., Gomes, C.P. (eds.) Theory and Applications of Satisfiability Testing - SAT 2006, 9th International Conference, Seattle, WA, USA, August 12-15, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4121, pp. 61–74. Springer (2006). https://doi.org/10.1007/11814948_9

5. Chavira, M., Darwiche, A.: On probabilistic inference by weighted model counting. Artif. Intell. **172**(6-7), 772–799 (2008). https://doi.org/10.1016/j.artint.2007.11.002

6. Chavira, M., Darwiche, A., Jaeger, M.: Compiling relational bayesian networks for exact inference. Int. J. Approx. Reason. **42**(1-2), 4–20 (2006). https://doi.org/10.1016/j.ijar.2005.10.001

7. Choi, A., Kisa, D., Darwiche, A.: Compiling probabilistic graphical models using sentential decision diagrams. In: van der Gaag, L.C. (ed.) Symbolic and Quantitative Approaches to Reasoning with Uncertainty - 12th European Conference, ECSQARU 2013, Utrecht, The Netherlands, July 8-10, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7958, pp. 121–132. Springer (2013). https://doi.org/10.1007/978-3-642-39091-3_11

8. Darwiche, A.: On the tractable counting of theory models and its application to truth maintenance and belief revision. J. Appl. Non Class. Logics **11**(1-2), 11–34 (2001). https://doi.org/10.3166/jancl.11.11-34

9. Darwiche, A.: A logical approach to factoring belief networks. In: Fensel, D., Giunchiglia, F., McGuinness, D.L., Williams, M. (eds.) Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning (KR-02), Toulouse, France, April 22-25, 2002. pp. 409–420. Morgan Kaufmann (2002)

10. Darwiche, A.: New advances in compiling CNF into decomposable negation normal form. In: de Mántaras, R.L., Saitta, L. (eds.) Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22-27, 2004. pp. 328–332. IOS Press (2004)

11. Darwiche, A.: SDD: A new canonical representation of propositional knowledge bases. In: Walsh, T. (ed.) IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011. pp. 819–826. IJCAI/AAAI (2011). https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-143, https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-143

12. Dudek, J.M., Phan, V., Vardi, M.Y.: ADDMC: weighted model counting with algebraic decision diagrams. In: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020. pp. 1468–1476. AAAI Press (2020), https://aaai.org/ojs/index.php/AAAI/article/view/5505

13. Dudek, J.M., Phan, V.H.N., Vardi, M.Y.: DPMC: weighted model counting by dynamic programming on project-join trees. In: Simonis, H. (ed.) Principles and Practice of Constraint Programming - 26th International Conference, CP 2020, Louvain-la-Neuve, Belgium, September 7-11, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12333, pp. 211–230. Springer (2020). https://doi.org/10.1007/978-3-030-58475-7_13

14. Oztok, U., Darwiche, A.: A top-down compiler for sentential decision diagrams. In: Yang, Q., Wooldridge, M.J. (eds.) Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015. pp. 3141–3148. AAAI Press (2015), http://ijcai.org/Abstract/15/443

15. Sang, T., Bacchus, F., Beame, P., Kautz, H.A., Pitassi, T.: Combining component caching and clause learning for effective model counting. In: SAT 2004 - The Seventh International Conference on Theory and Applications of Satisfiability Testing, 10-13 May 2004, Vancouver, BC, Canada, Online Proceedings (2004), http://www.satisfiability.org/SAT04/programme/21.pdf

16. Sang, T., Beame, P., Kautz, H.A.: Performing Bayesian inference by weighted model counting. In: Veloso, M.M., Kambhampati, S. (eds.) Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA. pp. 475–482. AAAI Press / The MIT Press (2005), http://www.aaai.org/Library/AAAI/2005/aaai05-075.php

17. Shen, Y.: Modeling, Learning and Reasoning with Structured Bayesian Networks. Ph.D. thesis, UCLA (2020)